On the average-case complexity of parameterized clique

Fountoulakis, Nikolaos; Friedrich, Tobias; Hermelin, Danny

DOI:
10.1016/j.tcs.2015.01.042

On the average-case complexity of parameterized clique

Nikolaos Fountoulakis, Tobias Friedrich, Danny Hermelin

PII: S0304-3975(15)00089-4
DOI: http://dx.doi.org/10.1016/j.tcs.2015.01.042
Reference: TCS 10053

To appear in: *Theoretical Computer Science*

Received date: 14 February 2013
Revised date: 23 October 2014
Accepted date: 23 January 2015

Please cite this article in press as: N. Fountoulakis et al., On the average-case complexity of parameterized clique, *Theoret. Comput. Sci.* (2015), http://dx.doi.org/10.1016/j.tcs.2015.01.042

On the Average-case Complexity of Parameterized Clique

Nikolaos Fountoulakis [a], Tobias Friedrich [b], Danny Hermelin [c]

[a] *University of Birmingham, Edgbaston, United Kingdom*

[b] *Hasso Plattner Institute, Potsdam, Germany*

[c] *Ben-Gurion University, Beer-Sheva, Israel*

Abstract

The k-Clique problem is a fundamental combinatorial problem that plays a prominent role in classical as well as in parameterized complexity theory. It is among the most well-known NP-complete and W[1]-complete problems. Moreover, its average-case complexity analysis has created a long thread of research already since the 1970s. Here, we continue this line of research by studying the dependence of the average-case complexity of the k-Clique problem on the parameter k. To this end, we define two natural parameterized analogs of efficient average-case algorithms. We then show that k-Clique admits both analogues for Erdős-Rényi random graphs of *arbitrary* density. We also show that k-Clique is unlikely to admit either of these analogs for some specific computable input distribution.

Key words: Parameterized complexity, computational complexity, average-case, clique

1 Introduction

The k-Clique problem is one of the most fundamental combinatorial problems in graph theory and computer science. This problem asks to determine whether a given graph contains a clique of size k, *i.e.* a complete subgraph on k vertices. The k-Clique problem forms the groundwork for many worst-case hardness frameworks: It is one of Karp's famous initial list of NP-complete problems [11], and its optimization variant is a classical example of a problem that is NP-hard to approximate within a factor of $n^{1-\varepsilon}$ for any $\varepsilon > 0$ [20]. In parameterized complexity theory [4], the k-Clique problem is textbook example complete for the class W[1], the parameterized analogue of NP, playing

a prominent role in W[1]-hardness results very much akin to the role 3-SAT plays in the classical complexity.

In this paper we are interested in the parameterized complexity of the k-Clique problem on "average" inputs. For our purposes, an average k-Clique instance can be naturally and conveniently modeled using the thoroughly-studied Erdős-Rényi distributions on graphs. The class of these distributions is typically denoted by $\mathcal{G}(n,p)$, with $n \in \mathbb{N}$ and $p \in [0,1]$, where on a graph with n vertices each pair of vertices are adjacent independently with probability p. Such random graphs have approximate density p, and it is well-known (see *e.g.* [1, 10]) that the typical properties of these random graphs are essentially the typical properties of a random graph that is uniformly selected among all graphs on n vertices and $p\binom{n}{2}$ edges.

The question of of finding cliques in $\mathcal{G}(n,p)$ random graphs has been raised by Karp [12] already in 1976. Karp observed that in $\mathcal{G}(n,1/2)$ (note that this is in fact the uniform distribution over all graphs on n vertices) the maximum size of a clique is about $2\log n$ with high probability, but the greedy algorithm only finds with high probability a clique that is approximately half this size. Karp asked whether in fact there is any polynomial-time algorithm that finds a clique of size $(1+\varepsilon)\log n$, for some $\varepsilon > 0$. This question remains open until today.

Finding cliques in $\mathcal{G}(n,p)$ random has also been considered when the clique sought after have small size, which is the main theme of our paper. For a fixed integer $k \geq 3$, the random graph $\mathcal{G}(n,p)$ undergoes a phase transition regarding the (almost sure) existence of cliques of size k (cf. [1] or [10]) as the edge probability p grows. More specifically, it is known that when $p \ll n^{-2/(k-1)}$, then $\mathcal{G}(n,p)$ does not contain any cliques of size k, with high probability, but when $p \gg n^{-2/(k-1)}$, then in fact there are many k-cliques with high probability. However, inside the "critical window", that is when $p = \Theta(n^{-2/(k-1)})$, the maximum size of a clique could be either $k-1$ or k each one occurring with probability that is bounded away from 0 as n grows to infinity. More precisely, the number of cliques of size k follows asymptotically a Poisson distribution with parameter that depends on k. In this range, the greedy algorithm finds a clique of size $\lfloor \frac{k}{2} \rfloor$ or $\lceil \frac{k}{2} \rceil$, with high probability. Rossman [17, Remark 35] remarks that repeating the greedy algorithm $n^{k/4+O(1)}$ times, we can enumarate all the maximal cliques with high probability. This gives a randomized algorithm with runtime $n^{k/4+O(1)}$ for solving k-Clique with high probability.

Since the above algorithm is the fastest algorithm known, it seems that a typical instance of $\mathcal{G}(n,p)$ with $p = \Theta(n^{-2/(k-1)})$ is in fact a hard instance for k-Clique. This is also suggested by the lower bounds on the size of monotone circuits for k-Clique derived recently by Rossman [17] (see also [16]) for p in this range. Thus any substantial improvement to the $n^{k/4+O(1)}$ algorithm

above would be a major breakthrough result; not to mention an FPT algorithm running in $f(k) \cdot n^{O(1)}$ time, which is perhaps far too much of an improvement than we can expect [1]. To avoid this obstacle, we consider distributions $\mathcal{G}(n,p)$ where p does not depend on k (but may depend on n). Apart from the obvious advantage that this gives a real chance at obtaining positive results, we also believe that this a very natural model of practical settings. Indeed, in many cases the distribution of the graphs we are interested in is fixed, while the size of the cliques we are looking for may vary.

We consider two types of algorithms running in FPT time on average. The first is an avgFPT-algorithm, which is an algorithm with expected $f(k) \cdot n^{O(1)}$ runtime. Thus, an avgFPT-algorithm is required to run in FPT-time on average according to the given input distribution. This means that the algorithm is allowed to be slow on some instances, so long as that its efficient on average. The notion of avgFPT-time is a natural parameterized analogue of an avgP-time algorithm (see *e.g.* [7]), and is perhaps the most natural definition of the notion "FPT on average".

We present a very simple avgFPT algorithm for k-Clique for essentially all distributions $p := p(n)$. By essentially, we mean all *natural* distributions that have typical properties, such as certain limit properties (this is made precise in Definition 5). The first result of this paper is thus the following theorem.

Theorem 1. *Let $p := p(n)$ denote a natural distribution function. There is an* avgFPT*-algorithm for k-*Clique *on graphs $G \in \mathcal{G}(n,p)$.*

The second type of average-case FPT algorithms we consider are algorithms that run in typical FPT (typFPT) time. By this we mean a running time of $f(k) \cdot n^{O(1)}$ with high probability, where high probability means that the algorithm is allowed to be slower only with probability smaller than any polynomial in n. Thus, one may view the difference between a typFPT-time algorithm and an avgFPT-time algorithm is that an avgFPT-time algorithm is allowed to be slightly slow on relatively many instances, while a typFPT-time algorithm is allowed to be extremely slow on relatively few instances. In stochastic terms, this is precisely the difference between bounding the expected value of a random variable and showing that it is bounded with high probability. Again, the analogous notion in classical complexity is typical P-time [7].

We show that the same algorithm used in Theorem 1 is actually a typFPT algorithm for k-Clique for any natural $p := p(n)$. However, the proof of this result is more involved than the former and requires a rather sophisticated tail bound argument.

[1] Note that $f(k) \cdot n^{O(1)} \ll n^k$ for any function f, when k is fixed and n tends to infinity.

Theorem 2. *Let $p := p(n)$ denote a natural distribution function. There is a* typFPT*-algorithm for* k-Clique *on graphs* $G \in \mathcal{G}(n,p)$.

It is worth mentioning that in both theorems above, our algorithms are completely deterministic and *always* correctly decide whether their input graph contains a clique of size k. This makes the proofs more challenging, since the algorithms cannot only assume that a k-clique is unlikely to exist in the input, but they must also certify this somehow. Furthermore, our algorithms can easily be modified to determining whether a $\mathcal{G}(n,p)$ random graph has an independent set of size k. Moritz Müller's PhD thesis [15] provides the first attempt at setting up a framework of parameterized average case complexity. In particular, he defined a notion very much similar to our avgFPT-algorithm, except that in his case the algorithm is allowed to have one-sides errors with constant probability. The notion of typFPT has not appeared elsewhere to the best of our knowledge. The distinction between these two types of average-case tractability notions is standard in the classical world, and in Section 2 we briefly argue why this distinction makes even more sense in the parameterized world. Müller also defined an average-case analogue of W[1], and showed that there is some (artificial) problem which is complete for it. We discuss this result in the last part of the paper, and show that the k-Clique problem is hard for this average-case analogue of W[1] on a specific distribution.

2 Average Case Parameterized Algorithms

In this section we define our two average-case analogues of FPT algorithms. We begin by some necessary terminology which follows the terminology used in Goldreich [7] for classical average-case analysis. A *distribution ensemble* X is an infinite sequence of probability spaces, one for each $n \in \mathbb{N}$, such that the n-th space is defined over $\{0,1\}^n$. We will associate with X a sequence of random variables $\{X_n\}_{n=1}^{\infty}$, where X_n is assigned strings in $\{0,1\}^n$ according to the corresponding distribution in X (thus, formally X_n maps strings from $\{0,1\}^n$ to strings from $\{0,1\}^n$). For example, we will write $\mathsf{Pr}[X_n = x]$ for the probability that X_n equals a specific $x \in \{0,1\}^*$ when drawn at random according to X. A *distributional parameterized problem* is a pair (L, X), where $L \subseteq \{0,1\}^* \times \mathbb{N}$ is a parameterized problem, and X is a distribution ensemble over strings in $\{0,1\}^*$.

Next let us consider avgFPT-time algorithms. Informally, we would like this class of algorithms to contain all algorithms running in FPT-time on average according to the distribution of their inputs. However, similar to the classical world, there are some technical problems with simply requiring that the corresponding algorithms run in expected FPT-time (*e.g.* this does not allow for robustness in the computation model, see [7]). Thus, as is done in the classical

setting, we will require some sort of *normalized* expected running time. Furthermore, we require that our algorithms always output the correct solution, or in other words, they must be able to decide the given problem.

Definition 3. *Let (L, X) be a distributional parameterized problem. We say that an algorithm $\mathcal{A}$ deciding L runs in* avgFPT*-time if there exists a constant c and a function $f\colon \mathbb{N} \to \mathbb{N}$ such that for all $k \in \mathbb{N}$:*

$$\sum_{n\in\mathbb{N}} \mathsf{E}\left[\frac{t_{\mathcal{A}}(X_n, k)}{n^c}\right] < f(k).$$

Here, and elsewhere, the random variable $t_{\mathcal{A}}(X_n, k)$ denotes the running time of an algorithm $\mathcal{A}$ on input (x, k), where x is chosen with probability $\mathsf{Pr}[X_n = x]$.

Observe that an avgFPT-time algorithm may run the brute-force procedure, which typically runs in $\mathcal{O}(n^k)$ time, with probability n^{-k}. This, as we will see further on, allows for a very simple analysis in some cases. A more stringent requirement of an efficient algorithm for parameterized distributional problems is to insist that it *typically* runs in FPT-time. That is, that it runs in FPT-time with high probability, where high probability means that the algorithm is allowed to be too slow only with probability super-polynomially small. Thus, a probability of n^{-k} will not suffice. This indicates that the distinction between the two average-case classes might be more apparent in the parameterized world than it is in classical complexity theory.

Definition 4. *Let (L, X) be a distributional parameterized problem. We say that an algorithm $\mathcal{A}$ deciding L runs in* typFPT*-time if there exists a function f and a polynomial p, such that for all $k \in \mathbb{N}$ and polynomials q there is an $n_0 \in \mathbb{N}$ such that for all $n > n_0$:*

$$\Pr[t_{\mathcal{A}}(X_n, k) > f(k) \cdot p(n)] < \frac{1}{q(n)}.$$

It is important to note that in the probability bound of the definition above we can equivalently use $f(k)/q(n)$ instead of $1/q(n)$. It is obvious that a $1/q(n)$ bound implies a $f(k)/q(n)$ bound (for $f(k) \geq 1$). To see the opposite direction, let us denote $\theta := \Pr[t_{\mathcal{A}}(X_n, k) > f(k) \cdot p(n)]$, and assume there exists a function f and polynomial p, such that for all parameters k and polynomials q there is an n_0 such that $\theta < f(k)/q(n)$ for all $n > n_0$. Then observe that at the time when the polynomial q is chosen, $f(k)$ is a fixed constant. Hence if $\theta < f(k)/q(n)$ holds for all polynomials q, then $\theta < f(k)/\tilde{q}(n)$ also holds for the polynomial $\tilde{q}(n)$ with $\tilde{q}(n) = f(k) \cdot q(n)$, which implies $\theta < 1/q(n)$ as required by Definition 4.

3 k-Clique is FPT on average

In this section we present an avgFPT-time algorithm for the k-Clique problem coupled with distribution ensembles defined via the Erdős-Rényi random graph model $\mathcal{G}(n,p)$ [5]. Recall that in $\mathcal{G}(n,p)$, a random graph on the vertex set $V := \{1,\ldots,n\}$, is constructed by connecting each pair of vertices independently with probability $p := p(n)$. We will show that for any *natural* function p, where the precise meaning of natural is given in Definition 5 below, there is an avgFPT-algorithm for k-Clique under $\mathcal{G}(n,p)$, providing the first part of the proof for Theorem 1.

Definition 5. *A function $p\colon \mathbb{N} \to [0,1]$ is natural if p either equals 0 for all $n \in \mathbb{N}$, or $p(n) := n^{-g(n)}$ for a non-negative function $g(n)$ where the limit $c_g := \lim_{n\to\infty} g(n)$ exists.*

The reader should observe that most commonly used functions p are natural or super-polynomially small [2]. For example, when $p(n) := 1/2$ we have $g(n) := 1/\lg n$ which is non-negative and $c_g = 0$, when $p(n) := 1/\lg n$ we have $g(n) = \lg\lg n/\lg n$, and for $p(n) := 1/n^c$ we have $g(n) = c$.

Our proof is split into two cases, one for dense graphs with $c_g = 0$ (Section 3.1), and the other for sparse graphs where $c_g > 0$ (Section 3.2). Clearly, showing that both the sparse and dense cases are in avgFPT shows that k-Clique is in avgFPT for all natural edge probabilities p.

Our algorithm is very simple in both the sparse and the dense case. In the dense case, with high probability we can find a k-clique among a linear number of k-subsets of vertices. If a solution is not found amongst these vertex subsets, we can exhaustively search through all k-subsets of vertices in the graph since this happens with very small probability. In the sparse case, we show that the expected number of maximal cliques is polynomial, and so we can use one of many algorithms (*e.g.* Tsukiyama, Ide, Ariyoshi, and Shirakawa [18]) to compute all maximal cliques in our input.

3.1 The dense case

Let $G \in \mathcal{G}(n,p)$ where $p := n^{-g(n)}$ with $c_g := \lim_{n\to\infty} g(n) = 0$, and n sufficiently large. Also, let $k \in \mathbb{N}$. Our algorithm for determining whether G has a k-clique, which we refer to as algorithm $\mathcal{A}$, is very simple: Let us call a clique of

[2] Note that for super-polynomially small p the k-Clique problem has trivial avgFPT and typFPT algorithms, since with super-polynomially high probability the input graph has no edges.

size k on a set of vertices $\{jk+1, \ldots, (j+1)\,k\} \subseteq V$, for $j \in \{0, \ldots, \lfloor n/k \rfloor - 1\}$, an *elementary k-clique*. Algorithm $\mathcal{A}$ first checks if G has an elementary k-clique. If so, it reports yes. Otherwise, it tries out all $\binom{n}{k}$ subsets of k vertices in G, reporting yes if and only if one of these is a clique.

It is clear that algorithm $\mathcal{A}$ correctly determines whether G has a k-clique in worst-case running-time $\mathcal{O}(k^2 n^k)$. Furthermore, as there are at most $\lfloor n/k \rfloor$ elementary k-cliques in G, checking whether elementary k-cliques are present in G requires $\mathcal{O}(k^2 n)$ time. Thus, if G contains an elementary k-clique, the running time of $\mathcal{A}$ is only $\mathcal{O}(k^2 n)$. The next lemma shows that for all interesting values of k, the probability that this event does not occur is exponentially small.

Lemma 6. *Let $k \leq \min\{n^{1/4},\, g(n)^{-1/4}\}$. Then*

$$\mathsf{Pr}[\,\mathcal{G}(n,p) \text{ contains no elementary } k\text{-clique}\,] \leq \exp\Big(-n^{1/2}\Big).$$

Proof. Let $EK(G)$ denote the number of elementary k-cliques in G. Observe that the probability that the vertex-subset $\{jk+1, \ldots, (j+1)k\} \subseteq V$, for a specific $j \in \{0, \ldots, \lfloor n/k \rfloor - 1\}$, is not a k-clique is $1 - p^{\binom{k}{2}}$, and this probability is independent of any other vertex-subset $\{j'k+1, \ldots, (j'+1)k\} \subseteq V$, $j' \neq j$, being a k-clique. Thus, using the fact that $\lfloor n/k \rfloor \geq n/k - 1 \geq n/(2k)$, we get for sufficiently large n:

$$\begin{aligned}\mathsf{Pr}[EK(G) = 0] = \left(1 - p^{\binom{k}{2}}\right)^{\lfloor n/k \rfloor} &\leq \exp\left(-\left\lfloor \frac{n}{k} \right\rfloor p^{\binom{k}{2}}\right) \\ &\leq \exp\left(-\frac{n}{2k}\, p^{\binom{k}{2}}\right) = \exp\left(-\frac{n^{1-g(n)\binom{k}{2}}}{2k}\right).\end{aligned}$$

Since $k \leq g(n)^{-1/4}$, we have $g(n)\binom{k}{2} \leq 1/4$ for sufficiently large n. Thus, since we also assume $k \leq n^{1/4}$, the right-hand side above can be bounded by $\exp\big(-n^{1/2}\big)$ for sufficiently large n. □

Lemma 6 gives us an easy way to bound the expected running-time of algorithm $\mathcal{A}$. Let $h(n) := g(n)^{-1/4}$. Observe that the worst case running-time of algorithm $\mathcal{A}$ is $\mathcal{O}(k^2 n^k)$. Let $h(n) := g(n)^{-1/4}$. Then $h(n)$ tends to infinity as n grows since $\lim_{n\to\infty} g(n)^{1/4} = 0$. Thus, for every k there exists a $\kappa(k)$ for which $k \leq h(n)$ for all $n \geq \kappa(k)$. If $n < \kappa(k)$, the worst-case running time of algorithm $\mathcal{A}$ can be bounded by $\mathcal{O}(k^2 n^k) = \mathcal{O}(k^2(\kappa(k))^k)$. This means that when $k > h(n) = g(n)^{-1/4}$ (and so $n \leq \kappa(k)$), the worst-case running-time of algorithm $\mathcal{A}$ can be bounded by a function in k. Similarly, if $k \geq n^{1/4}$, the worst-case running-time of $\mathcal{A}$ can also bounded by a function in k. Therefore, letting $f(k)$ denote a bound on the running-time of $\mathcal{A}$ in case

$k > \min\{n^{1/4},\, g(n)^{-1/4}\}$, we get by Lemma 6 above that

$$\begin{aligned}\mathsf{E}[t_{\mathcal{A}}(\mathcal{G}(n,p),k)] &= \mathcal{O}\Big(f(k) + \exp(-n^{1/2}) \cdot k^2 n^k + (1-\exp(-n^{1/2})) \cdot k^2 n\Big) \\ &= \mathcal{O}(f(k) \cdot n),\end{aligned}$$

and so

$$\sum_{n \in \mathbb{N}} \frac{\mathsf{E}\left[t_{\mathcal{A}}(\mathcal{G}(n,p),k)\right]}{n} = \mathcal{O}(f(k)),$$

proving that algorithm $\mathcal{A}$ runs in avgFPT-time.

3.2 The sparse case

Let $G \in \mathcal{G}(n,p)$ where $p := n^{-g(n)}$ with $c_g := \lim_{n\to\infty} g(n) > 0$, and let $k \in \mathbb{N}$. Our algorithm for this case, which we refer to as algorithm $\mathcal{B}$, is even simpler than algorithm $\mathcal{A}$: Algorithm $\mathcal{B}$ simply computes all maximal (with respect to set inclusion) cliques in G, using the classical algorithm of Tsukiyama et al. [18], and outputs yes if and only if one of the maximal cliques is of size at least k. Clearly, algorithm $\mathcal{B}$ correctly decides whether G has a k-clique.

The algorithm of Tsukiyama et al. [18] runs in $\mathcal{O}(n^3 MK(G))$ time, where $MK(G)$ denotes the number of maximal cliques in G. This is also the time complexity of algorithm $\mathcal{B}$. Thus, to bound the expected running time of $\mathcal{B}$ on $\mathcal{G}(n,p)$, it suffices to bound the expected number of maximal cliques that a graph in $\mathcal{G}(n,p)$ contains. To ease the analysis, we actually bound the number $K(G)$ of cliques in G, for which we always have $MK(G) \leq K(G)$.

For a graph G and a positive integer s, let $K_s(G)$ denote the number of cliques of size s in G. For any $s \geq 2$, the expected number of cliques of size s in $G \in \mathcal{G}(n,p)$ with $p = n^{-g(n)}$ is

$$\mu_s := \mathsf{E}\left[K_s(\mathcal{G}(n,p))\right] = \binom{n}{s} p^{\binom{s}{2}} \leq n^{s - g(n)\binom{s}{2}}. \tag{1}$$

Let $s_0 := 2\lceil \frac{4}{c_g} \rceil + 1$. If n is sufficiently large, then $g(n) > c_g/2$. A simple calculation then shows that if $s \geq s_0$, then $s - g(n)\binom{s}{2} \leq s - \frac{c_g}{2}\binom{s}{2} \leq -3s \leq -3$. Thereby, for any $s \geq s_0$ we have $\mu_s \leq n^{-3}$. Using this, we can easily bound $\mathsf{E}\left[K(\mathcal{G}(n,p))\right]$ for n large enough:

$$\mathsf{E}\left[K(\mathcal{G}(n,p))\right] = \sum_{s \geq 2} \mu_s = \sum_{s < s_0} \mu_s + \sum_{s \geq s_0} \mu_s \leq n^{s_0} + n \cdot n^{-3} \leq n^{s_0+1}.$$

Hence, the expected running time of $\mathcal{B}$ is $\mathcal{O}(n^{s_0+4})$, whence

$$\sum_{n\in\mathbb{N}} \frac{\mathsf{E}\left[t_{\mathcal{B}}(\mathcal{G}(n,p),k)\right]}{n^{s_0+4}} = \mathcal{O}(1),$$

shows that it indeed runs in avgFPT-time.

We want to point out that it is not hard to adjust the proof for the sparse case under the weaker assumption that the limit of $g(n)$ does not exist, but $0 < \liminf_{n\to\infty} g(n) < \limsup_{n\to\infty} g(n)$. However, if $0 = \liminf_{n\to\infty} g(n) < \limsup_{n\to\infty} g(n)$, then the density of the random graph varies substantially along appropriately chosen subsequences. In particular, one can find a subsequence over which the random graph has very slowly decaying density and another subsequence in which the random graph is sparse. In these cases, the proofs that are presented in this and the previous section can be applied over these subsequences. Thus, effectively one could combine the two algorithms into a single algorithm. However, such an algorithm would have expected running time which is far from the expected running time that one could achieve for dense random graphs.

4 k-Clique is typically FPT

In this section we argue that the k-Clique problem is in typFPT for all natural $\mathcal{G}(n,p)$ distributions, completing the proof of Theorem 1. As in Section 3, our proof will split into two cases: The dense case with $c_g = 0$, and the sparse case with $c_g > 0$, where c_g is the limit of the function $g(n)$ defining the edge-probability $p := n^{-g(n)}$. Moreover, the algorithms used in each case will be algorithms $\mathcal{A}$ and $\mathcal{B}$ of Section 3.

Observe that Lemma 6 shows that in the dense case with $c_g = 0$, algorithm $\mathcal{A}$ runs in $f(k) \cdot n$ time, with f as given in Section 3.1, with probability at least $1 - \exp(-n^{1/2})$. Thus, for dense edge probabilities, algorithm $\mathcal{A}$ runs in typFTP-time. The main challenge here is showing that algorithm $\mathcal{B}$ also runs in typFPT-time. Here, applying a simple tail bound such as Markov's inequality, allows us to show that algorithm $\mathcal{B}$ is too slow with only polynomially small probability. To show that it is in fact slow only with super-polynomially small probability requires a slightly more involved argument.

So let $p := n^{-g(n)}$ be such that $c_g := \lim_{n\to\infty} g(n) > 0$. Recall that the running-time of algorithm $\mathcal{B}$ on a graph G with n vertices is $\mathcal{O}(n^3 MK(G)) = \mathcal{O}(n^3 K(G))$. For an integer $s \geq 2$, we let $K_s(G)$ denote the number of cliques of size s in a graph G. Then $K(G) = \sum_{s=2}^{n} K_s(G)$. To bound $K(\mathcal{G}(n,p))$ with high probability, we show that there exists an $s_1 \in \mathbb{N}$ depending only on c_g

(and thus on p) such that with very high probability the total number of cliques of size at least s_1 in $\mathcal{G}(n,p)$ is at most logarithmic.

Lemma 7. *Let $p := p(n) := n^{-g(n)}$, with $g(n)$ such that $c_g := \lim_{n\to\infty} g(n) > 0$. Then there exists an $s_1 \in \mathbb{N}$ such that for any n sufficiently large with probability at least $1 - \exp(-n\log n)$, we have*

$$\sum_{s\geq s_1} K_s(\mathcal{G}(n,p)) \leq \log n.$$

Proof. We begin with giving a tail bound on the probability that $K_s(\mathcal{G}(n,p))$ is large for an arbitrary integer $s \geq 2$. Recall that by (1), for any such $s \geq 2$, the expected number μ_s of cliques of size s is bounded, for all $s \geq 2$, by $\mu_s \leq n^{s-g(n)\binom{s}{2}}$ for n sufficiently large. We now give an upper-tail bound on the number of cliques of size s in $\mathcal{G}(n,p)$ through which we will determine s. To this end, we will use an upper-tail inequality for sums of dependent random variables due to Janson and Ruciński [9]. Let $\mathcal{K}$ be a non-empty set and $\{X_S\}_{S\in\mathcal{K}}$ denote a family of non-negative random variables defined on the same probability space. For $S, S' \in \mathcal{K}$, we write $X_S \sim X_{S'}$ to denote that these random variables are dependent. For $S \in \mathcal{K}$, we let $\Delta_S := |\{S' \colon X'_S \sim X_S\}|$ and $\Delta = \max_{S\in\mathcal{K}} \Delta_S$. Assume also that for all $S \in \mathcal{K}$, we have $X_S \leq 1$. Now, let $X := \sum_{S\in\mathcal{K}} X_S$ and let $\mu := \mathsf{E}[X]$. Corollary 2.6 in [9] states that for any $t \geq 0$,

$$\mathsf{Pr}[X \geq \mu + t] \leq \left(1 + \frac{t}{\mu}\right)^{-\frac{t}{4\Delta}}. \tag{2}$$

In our application, the probability space is induced by the $\mathcal{G}(n,p)$ model of random graphs and $\mathcal{K}$ is the collection of all subsets of s vertices of G. For each such subset $S \in \mathcal{K}$, let $X_S \in \{0,1\}$ be the indicator random variable which equals 1 if and only if $G[S]$ is a clique. As far as the quantity Δ is concerned, for any $S \in \mathcal{K}$ with $|S| < n$ we have

$$\Delta_S = \sum_{i=2}^{s} \binom{s}{i}\binom{n-s}{s-i} \leq \sum_{i=2}^{s} s^i n^{s-i} = n^s \sum_{i=2}^{s} \left(\frac{s}{n}\right)^i \leq n^s \sum_{i=2}^{\infty} \left(\frac{s}{n}\right)^i \leq 2s^2 n^{s-2},$$

and therefore $\Delta \leq 2s^2 n^{s-2}$, as when $|S| = n$ then $\Delta_S = 0$. Since $\sum_{S\in\mathcal{K}} X_S = K_s(G)$ and letting $t = \log n/2s^2$, Inequality (2) yields

$$\begin{aligned}\mathsf{Pr}\left[K_s(G) \geq \mu_s + \log n/2s^2\right] &\leq \left(1 + \frac{\log n}{2s^2\mu_s}\right)^{-\frac{\log n}{8s^2\Delta}} \leq \exp\left(-\frac{n^{g(n)\binom{s}{2}}\log^2 n}{32 n^s s^6 n^{s-2}}\right) \\ &= \exp\left(-\frac{n^{g(n)\binom{s}{2}-2s+2}\log^2 n}{32s^6}\right). \end{aligned} \tag{3}$$

Now recall that $c_g = \lim_{n\to\infty} g(n) > 0$. Thus, for any n sufficiently large we

have $g(n) > 8c_g/10$. Since $s \geq 2$, we also have $\binom{s}{2} \geq s^2/4$, and therefore,

$$g(n)\binom{s}{2} - 2s + 2 > c_g \frac{s^2}{5} - 2s + 2.$$

Let us set $s_1 := \max\left\{\lceil \frac{25}{c_g} \rceil, 3\right\}$. We will show that for any $s \geq s_1$ we have $c_g s^2/5 - 2s_0 + 2 \geq 7$. That is, $s\,(c_g s/5 - 2) \geq 3$. Indeed, $s\,(c_g s/5 - 2) \geq s_1\,(c_g s_1/5 - 2) \geq s_1\,(5 - 2) > 7$.

As $s \geq s_1 \geq 3$, for n sufficiently large, this implies that $g(n)\binom{s}{2} - s > s - 1 \geq 2$, and therefore $\mu_{s_1} \leq n^{-2}$. Thus if n is sufficiently large, for all $s \geq s_1$ we have

$$\mathsf{Pr}\left[K_s(\mathcal{G}(n,p)) \geq \frac{\log n}{s^2}\right] \leq e^{-n\log^2 n/16}.$$

So applying the union bound we deduce that, if n is sufficiently large, with probability at least $1 - e^{-n\log n}$ we have

$$\sum_{s=s_1}^{n} K_s(\mathcal{G}(n,p)) \leq \log n \sum_{s=s_1}^{\infty} \frac{1}{s^2} \leq \log n. \qquad \square$$

Alternatively, we could derive a weaker bound with the use of large deviation inequalities for subgraph statistics in a random graph (see for example Theorem 2.2 in [19]).

The above lemma provides the existence of a constant s_1 depending on $g(n)$ such that for any n sufficiently large $K(\mathcal{G}(n,p)) \leq n^{s_1} + \log n$ with probability at least $1 - \exp(-n\log n)$. Thus the running time of algorithm $\mathcal{B}$ on sparse graphs is $\mathcal{O}(n^{s_1+3})$ with probability at least $1 - \exp(-n\log n)$, *i.e.*, it runs in typFPT-time.

5 A Hard Distribution for k-Clique

In the following section we show that there exists a certain distributional ensemble for which k-Clique coupled with this distribution is unlikely to have an avgFPT-algorithm, nor a typFPT-algorithm. We build on the theory developed by Müller [15], and use techniques developed in [8, 13] and [14] to prove our argument.

We begin by defining our average-case analogue of W[1]. A distribution en-

semble X is said to be *simple*[3] if there is a polynomial algorithm that on input $x \in \{0,1\}^*$, outputs the probability $\mathsf{Pr}[X_{|x|} \leq x]$, where $\leq$ denotes the standard lexicographic order on strings. In the classical world, the standard definition of the average-case analogue of NP is defined as all NP problems coupled with simple distributions. The restriction to simple distributions is done in order to avoid trivial hardness results. Thus, adapting the same line of discourse to the parameterized world, we define the class distW[1] as the set

$$\text{distW[1]} := \Big\{(L, X) \colon X \text{ is a simple distribution ensemble and } L \in \text{W[1]}\Big\}.$$

Note that this definition easily extends to any other parameterized class besides W[1]. The main working conjecture we propose for average-case parameterized analysis is $\text{distW[1]} \nsubseteq \text{avgFPT} \cup \text{typFPT}$.

We next define a reduction that preserves average-case parameterized tractability. The notion of a reduction we use here is essentially a hybrid of the two corresponding notions in classical average-case complexity and parameterized complexity.

Definition 8. *A distributional parameterized problem* (L_1, X) reduces *to another distributional parameterized problem* (L_2, Y)*, if there exists an algorithm* $\mathcal{A}$*, a function* f*, and a polynomial* p*, such that* $\mathcal{A}$ *on input* $(x, k) \in \Sigma^* \times \mathbb{N}$ *outputs in time* $f(k) \cdot p(|x|)$ *a pair* $(y, \ell) \in \Sigma^* \times \mathbb{N}$ *satisfying:*

- $(x, k) \in L_1 \iff (y, \ell) \in L_2$.
- $\ell \leq f(k)$.
- $|x| \leq |y|$.
- $\mathsf{Pr}[\mathcal{A}(X_{|x|}, k) = (y, \ell)] \leq f(k) \cdot p(|x|) \cdot \mathsf{Pr}[Y_{|y|} = y]$.

Observe that the first two requirements in Definition 8 are the usual requirements of a parameterized reduction. The third requirement is a technical requirement used also in non-parameterized distributional reductions that can typically be satisfied by a straightforward padding argument, yet it is necessary for the composition of our reductions (see Lemma 9). We note that this requirement is missing in Müller's work [15] since he was not interested in composing reductions. The last requirement, often referred to as the *domination property*, ensures that an infrequent input of L_1 does not get mapped to a frequent input of L_2. We let $(L_1, X) \leq (L_2, Y)$ denote the fact that (L_1, X) reduces, as per Definition 8, to (L_2, X).

Lemma 9. $\leq$ *is transitive.*

Proof. Let (L_1, X), (L_2, Y), and (L_3, Z) be three distributional parameterized

[3] Müller [15] uses here the term *polynomial-time distributed*

problems with $(L_1, X) \leq (L_2, Y)$ and $(L_2, Y) \leq (L_3, Z)$, and let $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively be the algorithms showing that $(L_1, X) \leq (L_2, Y)$ and $(L_2, Y) \leq (L_3, Z)$, as required by Definition 8. We prove that $(L_1, X) \leq (L_3, Z)$, by showing that the composition of $\mathcal{A}_2$ and $\mathcal{A}_1$ gives an algorithm that satisfies the conditions of Definition 8. It is easy to verify that the first three requirements of of Definition 8 hold. In particular, for any $(x, k) \in \Sigma^* \times \mathbb{N}$, the running-time of $\mathcal{A}_2(\mathcal{A}_1(x, k))$ (and hence, also its output size) is bounded by $f(k) \cdot p(|x|)$ for some computable $f()$ and polynomial $p()$, and moreover we have $m \leq f(k)$. To prove the lemma, we show that the probability that $\mathcal{A}_2(\mathcal{A}_1(X_{|x|}, k))$ outputs and $(z, m) \in \Sigma^* \times \mathbb{N}$ is bounded by above by the probability of z according to $Z_{|z|}$, modulo some FPT-factor in $|x|$ and k.

For this, note that all four requirements of Definition 8 for $\mathcal{A}_1$ and $\mathcal{A}_2$ hold with $f()$ and $p()$, and write

$$\mathsf{Pr}[\mathcal{A}_2(\mathcal{A}_1(X_{|x|}, k)) = (z, m)] \quad = \quad \sum_{\ell} \sum_{n} \sum_{\substack{y \text{ s.t } |y|=n, \\ \mathcal{A}_2(y,\ell)=(z,m)}} \mathsf{Pr}[\mathcal{A}_1(X_{|x|}, k)) = (y, \ell)].$$

Let n^* and ℓ^* denote the values of n and ℓ that maximize the rightmost sum above. Since there are only $f(k) \cdot p(|x|)$ choices for pairs (ℓ, n), we can restrict ourselves to bounding the rightmost sum above in terms of n^* and ℓ^*. By definition of $\mathcal{A}_1$, we have

$$\sum_{\substack{y^* \text{ s.t. } |y^*|=n^*, \\ \mathcal{A}_2(y^*,\ell^*)=(z,m)}} \mathsf{Pr}[\mathcal{A}_1(X_{|x|}, k)) = (y, \ell^*)] \;\leq\; f(k) \cdot p(|x|) \;\cdot \sum_{\substack{y \text{ s.t. } |y|=n^*, \\ \mathcal{A}_2(y,\ell^*)=(z,m)}} \mathsf{Pr}[Y_{n^*} = y].$$

Thus it suffices to bound the sum of probabilities in the rightmost sum above. Observe that this sum is precisely the probability that $\mathcal{A}_2(Y_{n^*}, \ell^*) = (z, m)$. By definition of $\mathcal{A}_2$, we get that

$$\mathsf{Pr}[\mathcal{A}_2(Y_{n^*}, \ell^*) = (z, m)] \;\leq\; f(\ell^*) \cdot p(n^*) \;\cdot\; \mathsf{Pr}[Z_{|z|} = z].$$

Now, recall that $\ell^* \leq f(k)$, and that $n^* = |y| \leq |z| \leq f(k) \cdot p(|x|)$ for every y as above (by the third requirement of Definition 8). Thus, $f(\ell^*) \cdot p(n^*) \leq f(f(k))p(f(k)) \cdot p(p(|x|))$, and the lemma is proven.

□

The next lemma shows the most important property of our reductions: For any pair of distributional parameterized problems (L_1, X) and (L_2, Y) with $(L_1, X) \leq L_2, Y)$, the question of whether (L_1, X) is tractable in the average-case parameterized sense reduces to same question regarding (L_2, Y). This has been shown for avgFPT-algorithms by Müller [15] [4]. We complement this re-

[4] In fact, [15] shows this for a more relaxed notion of reduction where the third

sult by showing that the same holds for typFPT-algorithms. For completeness, we also provide a proof for avgFPT in the appendix of the paper.

Lemma 10. *If* $(L_1, X) \leq (L_2, Y)$ *and* (L_2, Y) *has a* typFPT*-algorithm, then* (L_1, X) *also has a* typFPT*-algorithm.*

Proof. Let $\mathcal{A}$ be a typFPT algorithm for (L_2, Y) running in $f_{\mathcal{A}}(\ell) \cdot p_{\mathcal{A}}(|y|)$ time with high probability, and let $\mathcal{R}$ denote a reduction from (L_1, X) to (L_2, Y), as required by Definition 8, running in $f_{\mathcal{R}}(k) \cdot p_{\mathcal{R}}(|x|)$ time. We argue that the algorithm $\mathcal{B}$ which outputs $\mathcal{B}(x, k) := \mathcal{A}(\mathcal{R}(x, k))$ for all $(x, k) \in \Sigma^* \times \mathbb{N}$ is a typFPT-algorithm for (L_1, X). By definitions of $\mathcal{R}$ and $\mathcal{A}$, it is clear that $\mathcal{B}$ correctly decides (L_1, X). We show that algorithm $\mathcal{B}$ runs in more than $f(k) \cdot p(|x|)$ time with super-polynomially small probability, for $f()$ and $p()$ chosen such that $f(k) \cdot p(n)$ is sufficiently larger than $f_{\mathcal{R}}(k) \cdot p_{\mathcal{R}}(n) + f_{\mathcal{A}}(k) \cdot p_{\mathcal{R}}(n)$ for all k and sufficiently large n.

Fix $k \in \mathbb{N}$, and let $q()$ be an arbitrary polynomial. By our choice of $f()$ and $p()$, we can bound the the probability that $\mathcal{B}$ runs in more than $f(k) \cdot p(|x|)$ time by

$$\mathsf{Pr}[t_{\mathcal{B}}(X_{|x|}, k) > f(k) \cdot p(|x|)] \quad \leq \quad \sum_{\ell} \sum_{n} \sum_{\substack{y \text{ s.t. } |y|=n, \\ t_{\mathcal{A}}(y,\ell) > f_{\mathcal{A}}(\ell) \cdot p_{\mathcal{A}}(n)}} \mathsf{Pr}[\mathcal{R}(X_{|x|}, k) = (y, \ell)].$$

Note that there are at most $f_{\mathcal{R}}(k) \cdot p_{\mathcal{R}}(|x|)$ pairs of (ℓ, n) in the righthand side above. Thus, we can bound the total summation on the righthand side in terms of ℓ^* and n^* which are the values of ℓ and n that maximize the rightmost sum in this summation. Due to the requirements on $\mathcal{R}$, we get

$$\sum_{\substack{y \text{ s.t. } |y|=n^*, \\ t_{\mathcal{A}}(y,\ell^*) > f_{\mathcal{A}}(\ell^*) \cdot p_{\mathcal{A}}(n^*)}} \mathsf{Pr}[\mathcal{R}(X_{|x|}, k) = (y, \ell^*)] \quad \leq \quad f_{\mathcal{R}}(k) \cdot p_{\mathcal{R}}(|x|) \quad \cdot \sum_{\substack{y \text{ s.t. } |y|=n^*, \\ t_{\mathcal{A}}(y,\ell^*) > f_{\mathcal{A}}(\ell^*) \cdot p_{\mathcal{A}}(n^*)}} \mathsf{Pr}[Y_{n^*} = y].$$

Note that the rightmost sum is just the probability that $\mathcal{A}(Y_{n^*}, \ell^*)$ runs in more than $f_{\mathcal{A}}(\ell^*) \cdot p_{\mathcal{A}}(n^*)$ time. Since $\mathcal{A}$ is a typFPT-algorithm for (L_2, Y), this probability is super-polynomially small. In particular, it smaller than $1/q'(n)$, where $q'(n) := (f_{\mathcal{R}}(k) \cdot p_{\mathcal{R}}(n))^2 \cdot q(n)$. Note that $q'(n)$ is indeed a polynomial, as $p_{\mathcal{R}}()$ and $q()$ are polynomials, and $f(k)$ is fixed. Thus, we have

$$\begin{aligned} \mathsf{Pr}[t_{\mathcal{B}}(X_{|x|}, k) > f(k) \cdot p(|x|)] \quad &\leq \\ (f_{\mathcal{R}}(k) \cdot p_{\mathcal{R}}(|x|))^2 \quad \cdot \quad \mathsf{Pr}[t_{\mathcal{A}}(Y_{n^*}, \ell^*) > f_{\mathcal{A}}(\ell^*) \cdot p_{\mathcal{A}}(n^*)] \quad &\leq \\ \frac{(f_{\mathcal{R}}(k) \cdot p_{\mathcal{R}}(|x|))^2}{q'(|x|)} \quad &= \quad \frac{1}{q(|x|)}, \end{aligned}$$

requirement does not exist.

and the lemma is proven.

□

By distW[1]-*complete* we will mean, as usual, a problem $(L, X) \in \text{distW[1]}$ with $(L', Y) \leq (L, X)$ for every problem (L', Y) in distW[1]. Note that an avgFPT algorithm or a typFPT algorithm for a distW[1]-complete problem would falsify our working conjecture of $\text{distW[1]} \not\subseteq \text{avgFPT} \cup \text{typFPT}$. We therefore argue that showing that a problem is distW[1]-complete is strong evidence against the existence of such algorithms. In the remainder of the section we prove the following theorem:

Theorem 11. *Let L denote the k-*Clique *problem. There exists a simple distribution Y for which (L, Y) is* distW[1]*-complete.*

For proving Theorem 11, we need two initial results. The first states that there exists some (artificial) distW[1]-complete problem. This has been shown by Müller [15] using the same ideas as in [8, 13]. While Müller uses a slightly different notion of reduction than ours (his definition lacks the third requirement of Definition 8), his proof can easily be adopted to accommodate also our definition by a straightforward padding argument.

Theorem 12 ([15])**.** *There is a distributional parameterized problem (U, X) which is* distW[1]*-complete.*

The following lemma by Livne [14] (see also [7]) gives the necessary technical tool for reducing the (U, X) problem above to some distributional k-Clique. We assume some natural encoding of graphs into binary strings, and let $\langle G \rangle$ denote the encoding of a given graph G.

Lemma 13 ([14])**.** *There is a polynomial-time algorithm that given a graph G and an $x \in \{0,1\}^*$, computes a graph G_x such that:*

- *$x = x'$ and $G = G' \iff \langle G_x \rangle = \langle G_{x'} \rangle$.*
- *$|x| = |x'| \iff |\langle G_x \rangle| = |\langle G_{x'} \rangle|$.*
- *$|x| \leq |\langle G_x \rangle|$.*
- *G has a k-clique $\iff$ G_x has a k-clique, for any $k \neq 2$.*
- *If X is a simple distribution ensemble then the distribution ensemble Y defined by*

$$\mathsf{Pr}[Y_{|y|} = y] = \begin{cases} \mathsf{Pr}[X_{|x|} = x] : & y = \langle G_x \rangle \\ 0 : & y \neq \langle G_x \rangle \text{ for all } x \text{ and } \exists x \text{ s.t. } \langle G_x \rangle \in \{0,1\}^{|y|} \\ 1/2^{|y|} : & \text{otherwise } (\nexists x \text{ s.t. } \langle G_x \rangle \in \{0,1\}^{|y|}) \end{cases}$$

is also simple.

Proof of Theorem 11. Let (U, X) denote the distW[1]-complete problem of Theorem 12, and let L denote the k-Clique problem. Since $U \in$ W[1], and L is W[1]-complete, there exists a parameterized reduction $\mathcal{A}$ from U to L. We construct an alternative reduction $\mathcal{A}^*$ which works as follows:

(1) It first computes $\mathcal{A}(x, k) = (G, \ell)$.
(2) It then checks if $\ell = 2$:
  (a) If so, it sets $\ell^* := 3$ if G has no edges, and otherwise it sets $\ell^* := 1$.
  (b) If $\ell \neq 2$, it sets $\ell^* := \ell$.
(3) It then computes G_x, and outputs the pair (G_x, ℓ^*).

Clearly, $\mathcal{A}^*$ runs in FPT-time. Moreover, $\mathcal{A}^*$ is a reduction, as required by Definition 8, from (U, X) to (L, Y), where Y is the distribution defined in the last item of Lemma 13 above. Indeed, it is easy to see that

$$(x, k) \in U \iff (G, \ell) \in L \iff (G_x, \ell^*) \in L$$

by Lemma 13 and the definition of $\mathcal{A}$. Furthermore, since $\ell \leq f(k)$ for some f, we have $\ell^* \leq f(k) + 1$, and $|x| \leq |\langle G_x \rangle|$ by Lemma 13. Finally, by our construction and Lemma 13,

$$\mathsf{Pr}[\mathcal{A}^*(X_{|x|}, k) = (G_x, \ell^*)] = \mathsf{Pr}[X_{|x|} = x] = \mathsf{Pr}[Y_{|\langle G_x \rangle|} = \langle G_x \rangle].$$

Thus $(U, X) \leq (L, Y)$. Since Y is simple, $(L, Y) \in$ distW[1], and so by Lemma 9 we get that (L, Y) is distW[1]-complete. □

6 Discussion

In this paper we considered the average-case parameterized complexity of the fundamental k-Clique problem. We showed that when restricted to Erdős-Rényi random graphs of arbitrary density $p := p(n)$, the problem admits two types of natural average-case analogues of FPT algorithms: An avgFPT algorithm and a typFPT algorithm. Thus, in this sense, the worst-case W[1]-complete k-Clique problem is easy on average. Furthermore, by adaptation of arguments from classical average-case analysis due to Livne [14], it can also be shown that for specific distributions k-Clique is unlikely to be FPT on average (unless any problem in W[1] under any computable distribution is easy). k-Clique is also known to be easy for scale-free random graphs [6]. It would be interesting to characterize graph distributions for which k-Clique becomes easy, i.e., avgFPT or typFPT.

It would be interesting to see which other W[1]-hard problems are easy on Erdős-Rényi random graphs of arbitrary density $p := p(n)$. Here it is important to require that the algorithms are deterministic and always correct, to avoid trivial results. We remark that many of the arguments used for k-Clique do not seem to carry through easily to other problems. A particularly interesting case is the k-Dominating Set problem, the W[1]-hard problem of determining whether a given graph has a dominating set of size k. The hard instances for this problem seem to be $\mathcal{G}(n, 1/2)$.

We finally point out that studying the average-case behavior of W[1]-hard problems might not only be interesting for graph problems. Bringmann and Friedrich [2] study a variant of the well-known Klee's measure problem, which asks for the volume of a number of boxes in d-dimensional space. This problem is known to be W[1]-hard for the parameter d, but it becomes FPT on average if the input points are uniformly distributed on the standard simplex. We expect similar results to hold for other geometric W[1]-hard problems.

References

[1] B. Bollobás. *Random graphs*. Cambridge University Press, 2001.

[2] K. Bringmann and T. Friedrich. Parameterized average-case complexity of the hypervolume indicator. In *ACM Genetic and Evolutionary Computation Conference (GECCO)*, pages 575–582, 2013.

[3] Y. Chen, J. Flum, and M. Grohe. Bounded nondeterminism and alternation in parameterized complexity theory. In *18th Annual IEEE Conference on Computational Complexity (CCC)*, pages 13–29, 2003.

[4] R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[5] P. Erdős and A. Rényi. On random graphs. *Publ Math Debrecen*, 6: 290–297, 1959.

[6] T. Friedrich and A. Krohmer. Parameterized clique on scale-free networks. In *23rd International Symposium on Algorithms and Computation (ISAAC)*, pages 659–668, 2012.

[7] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[8] Y. Gurevich. Average case completeness. *Journal of Computer and System Sciences*, 42:346–398, 1991.

[9] S. Janson and A. Ruciński. The deletion method for upper tail estimates. *Combinatorica*, 4:615–640, 2004.

[10] S. Janson, T. Łuczak, and A. Ruciński. *Random graphs*. Wiley, 2000.

[11] R. Karp. Reducibility among combinatorial problems. In J. F. Traub, editor, *Complexity of Computer Computations*, pages 85–103. Academic Press, 1972.

[12] R. Karp. Probabilistic analysis of some combinatorial search problems. In *Algorithms and Complexity: New Directions and Recent Results*, pages 1–19, 1976.

[13] L. Levin. Average case complete problems. *SIAM Journal on Computing*, 15:285–286, 1986.

[14] N. Livne. All natural NP-complete problems have average-case complete versions. *Journal of Computational Complexity*, 19:477–499, 2010.

[15] M. Müller. *Parameterized Randomization*. PhD thesis, Albert-Ludwigs-Universität Freiburg im Breisgau, 2008.

[16] B. Rossman. *Average-Case Complexity of Detecting Cliques*. PhD thesis, Massachusetts Institute of Technology, 2010.

[17] B. Rossman. The monotone complexity of k-clique on random graphs. *SIAM Journal on Computing*, 43(1):256–279, 2014.

[18] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all the maximum independent sets. *SIAM Journal on Computing*, 6:505–517, 1977.

[19] V. H. Vu. A large deviation result on the number of small subgraphs of a random graph. *Combinatorics, Probability and Computing*, 10:79–94, 2001.

[20] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.

A Appendix

In this section we provide proofs for claims used in Section 5 which are proven in Müller's thesis [15] for definitions which are slightly different then ours. In particular we provide a proof for the avgFPT analog for Lemma 10, and a proof for Theorem 12. Our proofs here use the same techniques as in [15].

Lemma 14. *If* $(L_1, X) \leq (L_2, Y)$ *and* (L_2, Y) *has an* avgFPT*-algorithm, then* (L_1, X) *also has an* avgFPT*-algorithm.*

Proof. Let $\mathcal{A}$ be the algorithm as in Definition 3 showing that $(L_2, Y) \in$ avgFPT, and let $\mathcal{R}$ denote the reduction from (L_1, X) to (L_2, Y), as required by Definition 8. Also, let $f_{\mathcal{A}}$ and $p_{\mathcal{A}}$ be the computable function and polynomial associated with $\mathcal{A}$, and let $f_{\mathcal{R}}$ and $p_{\mathcal{R}}$ be the computable function and polynomial associated with $\mathcal{R}$. We show that the algorithm $\mathcal{B}$ which outputs $\mathcal{B}(x,k) := \mathcal{A}(\mathcal{R}(x,k))$ for all $(x,k) \in \Sigma^* \times \mathbb{N}$ gives a avgFPT algorithm for (L_1, X).

By definitions of $\mathcal{R}$ and $\mathcal{A}$, it is clear that $\mathcal{B}$ correctly decides (L_1, X). Furthermore, since for any $(x,k) \in \Sigma^* \times \mathbb{N}$, we have $t_{\mathcal{B}}(x,k) = t_{\mathcal{R}}(x,k) + t_{\mathcal{A}}(\mathcal{R}(x,k)) + O(1)$, by linearity of expectation, we have

$$\sum_{n\in\mathbb{N}} \mathsf{E}\left[\frac{t_{\mathcal{B}}(X_n,k)}{n^c}\right] \leq \sum_{n\in\mathbb{N}} \mathsf{E}\left[\frac{t_{\mathcal{R}}(X_n,k)}{n^c}\right] + \sum_{n\in\mathbb{N}} \mathsf{E}\left[\frac{t_{\mathcal{A}}(\mathcal{R}(X_n,k))}{n^c}\right],$$

for any $c \in \mathbb{N}$. As $t_{\mathcal{R}}(x,k) \leq f_{\mathcal{R}}(k) \cdot p_{\mathcal{R}}(|x|)$ for all $(x,k) \in \Sigma^* \times \mathbb{N}$, we have for any $k \in \mathbb{N}$

$$\sum_{n\in\mathbb{N}} \mathsf{E}\left[\frac{t_{\mathcal{R}}(X_n,k)}{n^c}\right] = \mathcal{O}\left(f_{\mathcal{R}}(k)\right)$$

for some sufficiently large c. Thus, to prove the lemma it suffices to bound the second summation above for every $k \in \mathbb{N}$.

Fix $k \in \mathbb{N}$. Due to the requirements on $\mathcal{R}$, we have for every $x \in \Sigma^*$

$$\begin{aligned}
\mathsf{E}[t_{\mathcal{A}}(\mathcal{R}(X_{|x|},k))] &= \textstyle\sum_\ell \sum_n \sum_{|y|=n} t_{\mathcal{A}}(y,\ell) \cdot \mathsf{Pr}[\mathcal{R}(X_{|x|},k) = (y,\ell)] \\
&\leq f_{\mathcal{R}}(k) \cdot p_{\mathcal{R}}(|x|) \cdot \textstyle\sum_\ell \sum_n \sum_{|y|=n} t_{\mathcal{A}}(y,\ell) \cdot \mathsf{Pr}[Y_n = y] \\
&= f_{\mathcal{R}}(k) \cdot p_{\mathcal{R}}(|x|) \cdot \textstyle\sum_\ell \sum_n \mathsf{E}[t_{\mathcal{A}}(Y_n,\ell)].
\end{aligned}$$

Now observe, that the number of summands on the right-hand side of the above inequality is finite, and, therefore, there exist n^*, ℓ^* that maximize the summands. In particular, observe that the number of summands is at most

$f_{\mathcal{A}}(k) \cdot p_{\mathcal{A}}(n)$. Thus,

$$\mathsf{E}[t_{\mathcal{A}}(\mathcal{R}(X_{|x|}, k))] \leq f_{\mathcal{R}}(k) \cdot f_{\mathcal{A}}(k) \cdot p_{\mathcal{R}}(|x|) \cdot p_{\mathcal{A}}(|x|) \cdot \mathsf{E}[t_{\mathcal{A}}(Y_{n^*}, \ell^*)].$$

But $n^* \leq f_{\mathcal{A}}(k) \cdot p_{\mathcal{A}}(n)$, which, in turn, implies that for any $c > 0$ we have

$$(n^*)^c \leq (f_{\mathcal{A}}(k) \cdot p_{\mathcal{A}}(|x|))^c \, |x|^c.$$

Thus, for any positive c we have

$$\mathsf{E}\left[\frac{t_{\mathcal{A}}(\mathcal{R}(X_{|x|}, k))}{|x|^c}\right] \leq \frac{f_{\mathcal{R}}(k) \cdot f_{\mathcal{A}}(k)}{f^c_{\mathcal{A}}(k)} \; \frac{p_{\mathcal{R}}(|x|) \cdot p_{\mathcal{A}}(|x|)}{p^c_{\mathcal{A}}(|x|)} \; \mathsf{E}\left[\frac{t_{\mathcal{A}}(Y_{n^*}, \ell^*)}{(n^*)^c}\right].$$

As we need to take the sum of the above over all $n \in \mathbb{N}$, observe that on the right-hand side the same value of n^* can be repeated at most n^* times. Thus, we obtain

$$\begin{aligned}\sum_{n \in \mathbb{N}} \mathsf{E}\left[\frac{t_{\mathcal{A}}(\mathcal{R}(X_n, k))}{n^c}\right] &\leq \frac{f_{\mathcal{R}}(k) f^c_{\mathcal{A}}(k)}{f^c_{\mathcal{A}}(k)} \; \sum_{n \in \mathbb{N}} n \; \frac{p_{\mathcal{R}}(n) p_{\mathcal{A}}(n)}{p^c_{\mathcal{A}}(n)} \; \mathsf{E}\left[\frac{t_{\mathcal{A}}(Y_n, \ell^*(n))}{n^c}\right] \\ &\leq \frac{f_{\mathcal{R}}(k)}{f^{c-1}_{\mathcal{A}}(k)} \; \sum_{n \in \mathbb{N}} n \; \frac{p_{\mathcal{R}}(n)}{p^{c-1}_{\mathcal{A}}(n)} \; \mathsf{E}\left[\frac{t_{\mathcal{A}}(Y_n, f_{\mathcal{A}}(k))]}{n^c}\right].\end{aligned}$$

Choosing c large enough, concludes the proof of the lemma.

□

Before providing the proof of Theorem 12, we need to describe the machine characterization for W[1] of Chen, Flum, and Grohe [3]. The characterization is based on a nondeterministic version of *random access machines* (RAM) which are a more accurate model of real-life computation than Turing machines. A RAM consists of an infinite set of registers $\{r_0, r_1, r_2, \ldots\}$, a program counter x, and an instruction set. The instructions are of the form `STORE` i or `ADD` i, j, and so forth (see [3] for details). A *nondeterministic RAM* (NRAM) consists of an additional instruction of the form `GUESS` i, j, which results in the machine "guessing" a number less than or equal to the number stored in register r_i, and storing this number in r_j [3]. Chen et al. used the following type of NRAM programs to characterize W[1]:

Definition 15. *A* NRAM *program P is a* W[1]*-program if there exists a computable function f and a polynomial p such that on every input (x, k), the program P on every run*

- *performs at most $f(k) \cdot p(|x|)$ instructions, storing numbers which are $\leq f(k) \cdot p(|x|)$ only in the first $f(k) \cdot p(|x|)$ registers;*
- *in every run of P, all nondeterministic instructions are among the last $f(k)$*

instructions of the computation.

In this case, we say that P accepts (x, k) using $(f(k), p(|x|))$ resources.

Theorem 16 ([3])**.** *A parameterized problem L is in* W[1] *iff there exists a* W[1]*-program P deciding L.*

Theorem 16 above suggests the following universal problem U for W[1]: Given an NRAM program P, an input $(x, \ell) \in \Sigma^*$, a unary integer t, and a parameter k, decide whether P accepts (x, ℓ) using (t, k) resources. It is clear that U is in W[1]: On input $(\langle P, (x, \ell), t\rangle, k)$, a W[1]-program Q can simulate, using $(\mathcal{O}(t), \mathcal{O}(k))$ resources, all runs of P on (x, ℓ) that use (t, k) resources. We next define a simple uniform distribution ensemble Y for U given by

$$\mathsf{Pr}[Y_n = \langle P, (x, \ell), t\rangle] := \frac{1}{2^{|P|+|x|} \cdot (\ell + t)},$$

where $n := |P| + |x| + \ell + t$. It is not difficult to verify that under a suitable encoding of NRAM programs, the above distribution is simple. Thus, $(U, Y) \in$ distW[1]. We will show that (U, Y) is in fact distW[1]-complete, using the following lemma initially proved by Levin [13].

Lemma 17 ([13])**.** *Let X be a simple distribution ensemble. Then there exists a polynomial-time computable, and polynomial-time invertible, injective function $\Psi\colon \Sigma^* \to \Sigma^*$, such that for all $x \in \Sigma^*$ we have $\mathsf{Pr}[X_{|x|} = x] \leq 2^{-(|\Psi(x)|+1)}$.*

Proof of Theorem 12. Let (L, X) be a problem in distW[1]. We reduce (L, X) to (U, Y) by mapping an instance $(x, k) \in \{0, 1\}^* \times \mathbb{N}$ to an instance $(\langle P, (x', k), t\rangle, \ell)$ as follows: Denote by Ψ the function given in Lemma 17, and let p_Ψ be the polynomial bounding the running-time of computing and inverting Ψ. Since $(L, X) \in$ distW[1], $L \in$ W[1], and so by Theorem 16 there is a W[1]-program Q deciding L. Let f_Q and p_Q denote the computable function and polynomial associated with Q as in Theorem 16. Define P to be the program that gets $x' := \Psi(x)$ as input, computes $x = \Psi^{-1}(x')$, and then simulates Q on (x, k) (accepting iff Q accepts). Finally, define $t := p_\Psi(|x'|) + p_Q(|x| + \ell) + c$, where c is the overhead time required to simulate Ψ^{-1} and Q, and let $\ell := f_Q(k)$.

Observe that our construction can be carried out in FPT-time, since writing down P is done in time independent of (x, k). Furthermore, clearly $\ell \leq f_Q(k)$, and since Q decides L, we have $(x, k) \in L \iff (\langle P, (x', k), t\rangle, \ell) \in U$. Thus, the first two requirements of Definition 8 are satisfied by the construction. The third requirement can be satisfied by padding P as necessary. Finally, to see that the last requirement is also satisfied, observe that the probability of

$y := \langle P, (x', k), t\rangle$ in Y is at least

$$\mathsf{Pr}[Y_{|y|} = y] := \frac{1}{2^{|P|+|\Psi(x)|} \cdot (k+t)} \geq \frac{1}{c' \cdot |y|} \cdot \frac{1}{2^{|\Psi(x)|}},$$

where c' is a constant depending only on P and Ψ, and not on (x, k). On the other hand, according to Lemma 17 we have

$$\mathsf{Pr}[X_{|x|} = x] \leq \frac{1}{2^{|\Psi(x)|+1}}.$$

Thus, by letting p denote the polynomial $p(n) := c'n/2$, combining these two inequalities gives

$$\mathsf{Pr}[X_{|x|} = x] \leq p(|y|) \cdot \mathsf{Pr}[Y_{|y|} = y].$$

Noting that (x, k) is the only pair that gets mapped to (y, ℓ) by our construction, the theorem follows. □